|  |
| --- |
| CSCE 401: Software Engineering |
| Saving Face |
| Facial Recognition using the Intel Creative Camera |
|  |
| Andrew Mason ([aemason@alaska.edu](mailto:aemason@alaska.edu))  Keith Schneider ([1900Geek@gmail.com](mailto:1900Geek@gmail.com))  Jacob Dempsey ([Jacob@JacobsChaos.com](mailto:Jacob@JacobsChaos.com)) |
| **12/2/2013** |

|  |
| --- |
| In our project, Saving Face, the Intel Creative Camera’s depth sensor is used to collect data points from which a 3D model is produced, and used for future comparison to identify users. This report covers the development process, the algorithms used, and code implementation. |

Contents

[Overview of Requirements and Features: 1](#_Toc373832022)

[Intel Creative Camera Specifications 1](#_Toc373832023)

[Recommended System Configuration 1](#_Toc373832024)

[Feature List: 2](#_Toc373832025)

[Development Estimates and Actual Progress: 3](#_Toc373832026)

[Description of the Design: 4](#_Toc373832027)

[Generic Algorithm for the model construction 4](#_Toc373832028)

[Generic Algorithm for the Model Comparison 4](#_Toc373832029)

[Description of the Finished Product: 5](#_Toc373832030)

[Data Collection 5](#_Toc373832031)

[Model Creation 5](#_Toc373832032)

[Model Storage 5](#_Toc373832033)

[Model Comparison 5](#_Toc373832034)

[Description of Tests: 5](#_Toc373832035)

[Description of the Process: 5](#_Toc373832036)

[Who Worked on What: 6](#_Toc373832037)

[Keith Schneider 6](#_Toc373832038)

[Andrew Mason 6](#_Toc373832039)

[Jacob Dempsey: 6](#_Toc373832040)

[Overall Experience: 6](#_Toc373832041)

[References 7](#_Toc373832042)

# Overview of Requirements and Features:

Our project requires the use of an Intel Creative Camera (*Figure 1*) for the collection of three dimensional data points used in all of the functionality of our project.

Figure : Intel Creative Camera.

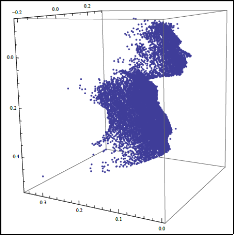
Saving Face features 3D model reconstruction and comparison of a person’s face (Figure 2). The comparison is performed by mapping retrieved depth values to a 3D model based off of the yaw, pitch, and roll of the face centering the tip of the user’s nose to the origin of the 3D coordinate system (Figure 3). The model data is collected over a predefined number of frames to give a better estimate of the user’s face in the model. The user’s initial model, or reference model, is stored in a database, and the identification models are then compared to the models in the database to find the closest match.

Figure : 3D model data.

## Intel Creative Camera Specifications

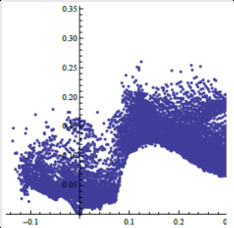
Intel® Developer Kit (2013) 

Figure : data centered at the origin.

Maximum RGB Video Resolution: HD 720p (1280x720)

Maximum IR Depth Resolution: QVGA (320x240)

Frame Rate: 30 fps

FOV (Diagonal): 73°

Range: 0.5ft to 3.25ft

Size: 4.27”x2.03”x2.11”

Weight: 9.56 oz

Power: Single USB 2.0 (power < 2.5W)

Dual-array microphones

RGB + Depth frame sync

## Recommended System Configuration

Intel® Developer Kit (2013)

PC with 2nd Generation or newer Intel® Core™ processor

Windows® 7 with Service Pack 1 or higher

4 GB system memory

USB 2.0 port

## Feature List:

|  |  |  |
| --- | --- | --- |
| **Story Name** | **Description** | **NUTS** |
| Fixed Point Orientation | Find X Y Z of at least three fixed points; Or 1 and Yaw Pitch and Roll | 2 |
| Transform Module | A Module that applies the Transform to bring the points into a common plane | 6 |
| Build Model Module | A Module that takes the transformed data and makes a model representation | 5 |
| Compare Module | A Module that takes a stream and compares it to a list of Models and returns a ranking | 5 |
| Load Module | Load a model from a file | 2 |
| Save Module | Saves a model to a file | 2 |
| Debug Terminal | Displays Helpful Information | 2 |
| Save Depth and Color Video | A Module that saves video for later re-use | 4 |
| Display Streams | A Module that displays Depth and Color Streams | 1 |
| User Information Input | A GUI Interface for inputting model Information | 3 |
| Save Image Data | A Module To Save a User Photo for Later Display | 2 |
| Load Color/Depth Video | A Module To Load Video Playback for Algorithm Refinement and Debugging | 4 |
| Load user Image | A Module to Load and display a user image | 2 |
| Activity: Model Collection | Collect Model Data To Test | 3 |

# Development Estimates and Actual Progress:

For most sections of our project, the estimates were relatively accurate. Toward the end, some were altered by a few points. Our burn up chart (*Figure 4*) accurately represents the team’s completion of nuts during this process. Our project had iterations that were not truly defined by iterations of equal amounts of work time so it is hard to encapsulate the iterations in the chart. Our first iteration may have had 10 – 15 actual hours of work, where as our last iteration easily had over 80 hours of work. From the time that the project started to now we noticed an exponential increase of hours put in vs. time passed. The burn up chart increases as we noticed we had to revisit a previous user story, and we switched from a debug terminal to a GUI. There was also a function that was not implemented in the PC SDK that we were relying on. In realization that this method was not implemented we had to create our own function to convert color coordinates to depth coordinates.

Our estimates mid semester showed us that we were not going to finish the project on time. However thanks to complete devotion at this critical point we tripled the amount of work being done. The estimates taken several weeks after showed us that we could finish the project on time.

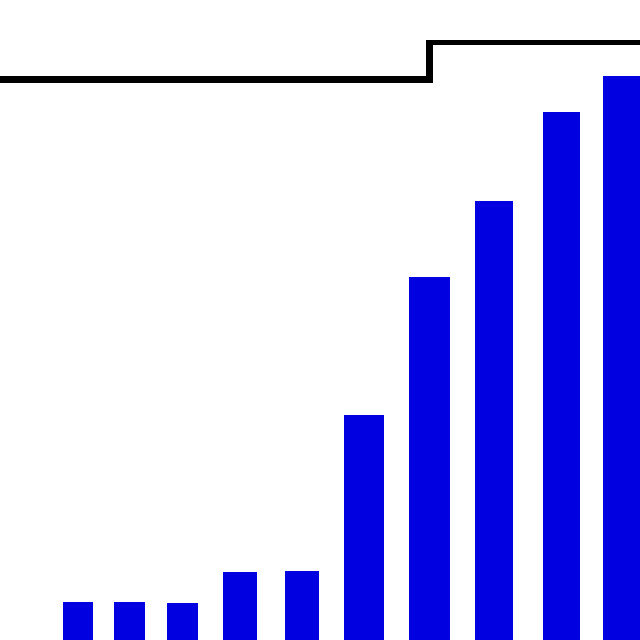


Figure : Saving Face burn up chart.

# Description of the Design:

Keith Schneider had some great foresight into the project. He designed early prototypes of the algorithms we needed to align the depth images for all models into a byte array. The functions used output from the Intel Creative Camera that was loaded into Mathematica for testing. Here is some pseudo code of the model building algorithm:

## Generic Algorithm for the model construction

While[Frames < N]

Read in byte stream

Get Fixed Point, and Yaw Pitch and Roll From SDK

Calculate Transformation Matrix Based On Yaw Pitch and Roll

Determine Relative Location Of Head

For[Vertices]

Apply Linear Transform to bring point relative to origin

Apply Rotational Transform To Align Head

Map Point to Model Byte[]

Increase that point by one

Save Model

End

In order to compare models it's essential that all of the models line up the same in our vertices array. Using the yaw, pitch, and roll we calculated a transformation matrix that was applied to all vertices to make all models the same rotation. We also centered the nose of each model at the origin of the 3D plane. With our models not only in the same rotation but also in the same position, we are able to compare points from our user model with points of the models in our database. The more points that match, the more likely the user is the particular model in the database.

## Generic Algorithm for the Model Comparison

CurrentMatch

BestMatch

For[each model in the database]

Vertex = 0

Match = 0

For[i=0; Vertices < N; i++]

If[identification.vertex[i]] // contains a data point

Vertex++ // count of vertices

If[reference.vertex[i]] // contains a data point

Match++ // count of matches CurrentMatch.ID = reference.ID

CurrentMatch.percentage = Match / Vertex

If[BestMatch.percentage < CurrentMatch.percentage]

BestMatch = CurrentMatch

End

# Description of the Finished Product:

Saving Face is an application designed to build and compare 3D models using the Intel Creative Camera. Saving Face takes depth images of a users face and builds a 3D model of them. The software is designed to test the feasibility of using the Intel Creative Camera for facial recognition.

## Data Collection

When creating a new model for the database, the use is asked for basic information used to facilitate identification and for contacting the user to ask them to participate in future tests. Information collected includes the users preferred salutation; their first, middle, and last name; a suffix if used; the users email address; and the persons gender. A picture of the user is then taken and stored with the model for human comparison after the software has selected a match after comparisons have been performed.

## Model Creation

The software captures and combines a set number of depth frames from the camera into a three dimensional model of the users face. The yaw, pitch, and roll of the users head, as well as the tip of the user’s nose are then calculated. The yaw, pitch, roll, and tip of the nose are then used to calculate a transform matrix to translate the model so the nose is centered at the origin. The model data is stored in a byte array with the vertices stored in a range versus a fixed point.

## Model Storage

When creating a new model for the database, the model, associated data, and a unique identifier are stored in a database for convenient storage and access.

## Model Comparison

The software creates an identification model of the user and compares the identification model to the reference models stored in the database. Each vertex area of the identification model is compared to the vertexes areas of the reference models. The number of matches are recorded and divided by the number of vertexes attempted to give a matching percent. The higher the percent match, the more likely the user was found in the database. A percentage threshold for a positive identification has not been established at this time.

# Description of Tests:

Add anything you want here, there was more to testing than just this

We used an automated testing system for the development process, we did do testing first but sometimes we did testing after. The automated testing system was an excellent way to track bugs and fix them. Testing first gave us more insight on how to avoid bugs in a given function. Testing first may feign a delay of progress but it saved so much time in the long run. We have multiple tests for functions and did a good job at determining a function was working based on an correct output with given inputs.

# Description of the Process:

We started the project using the Agile Method. We met with the client, created user stories, estimated NUTS for each store and noted dependencies some of the stories had with other stories. We then met with our client again to set priorities for the stories, and began working on those stories. As the work progressed, particularly when we realized we were not going to be able to complete the project based on our current number of hours being dedicated to the project, our client meetings became more informal short visits with our client as time permitted. The informal meetings were typically immediately after our client finished lecturing to his CSCE 401 class. We wrote and performed functionality tests on each section of the code as we progressed in the project, which proved invaluable in finding bugs in our implementation. However, toward the end of the project, we placed too much emphasis on completing the project; and began moving on to new user stories without first performing the corresponding client acceptance tests. With time constraints at our backs, we had to make the largest jumps we could in the shortest amount of time.

# Who Worked on What:

## Keith Schneider

Keith stepped up early in the project as the project lead. Throughout the project stayed on point as the project analyst, project lead, concept designer, lead programmer and lead tester.

## Andrew Mason

Andrew laid the early ground work for the GUI in C#, added ideas and insights to the design and algorithms and functioned as an assistant programmer, assistant tester, and assistant designer.

## Jacob Dempsey:

Jacob began implementing the GUI in C++ using MFC Assistant programmer and lead GUI designer.

# Overall Experience:

I think each of us should throw some of our input into this section of the report.

One of the biggest challenges was creating a constant schedule for our project. While other classes interfered we found that our hours of work varied week by week. Creating a definitive schedule of what needed to be done and how many hours should be done in an iteration for each team member would have helped a lot. We learned many new programming techniques and also a new and wonderful SDK. This was a very large project and in hind sight we believe that more team meetings to get the team organized would have been be better. Allowing more direction for individual tasks and also adding a better understanding of the code implementation.

# References

Intel Corporation. (2013) “Creative Intel Developer Kit”. Retrieved from http://download-software.intel.com/sites/default/files/article/325946/creativelabs-camera-productbrief-final.pdf December 2, 2013.